Workforce Scheduling Using the PEAST Algorithm

Nico R. M. Kyngäs, Kimmo J. Nurmi and Jari R. Kyngäs

Abstract Workforce scheduling has become increasingly important for both the public sector and private companies. Good rosters have many benefits for an organization, such as lower costs, more effective utilization of resources and fairer workloads and distribution of shifts. This paper presents a framework and an algorithm that have been successfully used to model and solve workforce scheduling problems in Finnish companies. The algorithm has been integrated into market-leading workforce management software in Finland.

Keywords Computational intelligence · Local search · Metaheuristics · PEAST algorithm · Population-based methods · Staff scheduling · Workforce scheduling

1 Introduction

Workforce scheduling, also called staff scheduling and labor scheduling, is a difficult and time consuming problem that every company or institution that has employees working on shifts or on irregular working days must solve. The workforce scheduling problem has a fairly broad definition. Most of the studies focus on assigning employees to shifts, determining working days and rest days or constructing flexible shifts and their starting times. Different variations of the problem and subproblems are NP-hard and NP-complete [1–5], and thus extremely hard to solve. The first mathematical formulation of the problem based on a generalized set covering model was

N. R. M. Kyngäs (✉) · K. J. Nurmi · J. R. Kyngäs
Satakunta University of Applied Sciences, Tiedepuisto 3, 28600 Pori, Finland
e-mail: nico.kyngas@samk.fi

K. J. Nurmi
e-mail: cimmo.nurmi@samk.fi

J. R. Kyngäs
e-mail: jari.kyngas@samk.fi

proposed by Dantzig [6]. Good overviews of workforce scheduling are published by Alfares [7], Ernst et al. [8] and Meisels and Schaerf [9].

Section 2 briefly introduces the necessary terminology and the workforce scheduling process as we have encountered it in various real-world cases. In Sect. 3 we describe the preprocessing phase of the workforce scheduling process. Section 4 presents the staff scheduling phase. Along with the problems and definitions of the subphases themselves we introduce some new real-world cases. Section 5 gives an outline of our computational intelligence algorithm.

We have used the PEAST algorithm, as described in Sect. 5, to solve numerous real-world staff scheduling problems for different Finnish companies. The algorithm has been integrated into the workforce management system of our business partner, and it is in constant real-world use.

2 Terminology and the Workforce Scheduling Process in Brief

The *planning horizon* is the time interval over which the employees have to be scheduled. Each employee has a total working time that he/she has to work during the planning horizon. Furthermore, each employee has *competences* (qualifications and skills) that enable him/her to carry out certain tasks. Days are divided into *working days* (days-on) and *rest days* (days-off). Each day is divided into periods or timeslots. A *timeslot* is the smallest unit of time and the length of a timeslot determines the granularity of the schedule. A *shift* is a contiguous set of working hours and is defined by a day and a starting period on that day along with a *shift length* (the number of occupied timeslots) or *shift time*. Shifts are sometimes grouped into *shift types*, such as morning, day and night shifts. Each shift is composed of *tasks* and *breaks*. The sum of the length of a shift's tasks is called *working time*, whereas sometimes the sum of the length of a shift's breaks is called *linkage time*. A timeslot-long piece of a task or break is called an *activity*. A consecutive sequence of activities dedicated to a single task is called a *stretch*. A shift or a task may require the employee assigned to it to possess one or more *competences*. A work schedule over the planning horizon for an employee is called a *roster*. A roster is a combination of shifts and days-off assignments that covers a fixed period of time.

Workload prediction, also referred to as demand forecasting or demand modeling, is the process of determining the staffing levels—that is, how many employees are needed for each timeslot in the planning horizon. The staffing is preceded by actual workload prediction or *workload determination* based on static workload constraints given by the company, depending on the situation. In *preference scheduling*, each employee gives a list of preferences and attempts are made to fulfill them as well as possible. The employees' preferences are often considered in the days-off scheduling and staff rostering subphases, but may also be considered during shift generation. Together these two subphases form the *preprocessing phase*.

Shift generation is the process of determining the shift structure, along with the activities to be carried out in particular shifts and the competences required for different shifts. *Days-off scheduling* deals with the assignment of rest days between working days over a given planning horizon. Days-off scheduling also includes the assignment of vacations and special days, such as union steward duties and training sessions. *Staff rostering*, also referred to as shift scheduling, deals with the assignment of employees to shifts. It can also specify the starting time and duration of shifts for a given day, even though in most cases they are pre-assigned during shift generation. This subphase may include both *resource analysis* to examine the compatibility between the available workforce and the shifts, and *partitioning* in the case of massive datasets (i.e. hundreds of employees) to speed up and improve on the results of the rostering. Together these five subphases form the *staff scheduling phase*.

Rescheduling deals with ad hoc changes that are necessary due to sick leaves or other no-shows. The changes are usually carried out manually. Finally, participation in *evaluation* ranges from the individual employee through personnel managers to executives. A reporting tool should provide performance measures in such a way that the personnel managers can easily evaluate both the realized staffing levels and the employee satisfaction. When necessary, parts of the whole workforce scheduling process may be restarted. *Workforce scheduling* consists roughly of everything from determining the needs of the customers to determining the exact schedule of each employee.

The workforce scheduling process presented in this paper is mostly concerned with short-term planning, as defined in [10]. We have chosen to split the problem into subphases as seen in Fig. 1. This may cause problems in extremely difficult cases, due to the search space at each subphase being constrained by the choices

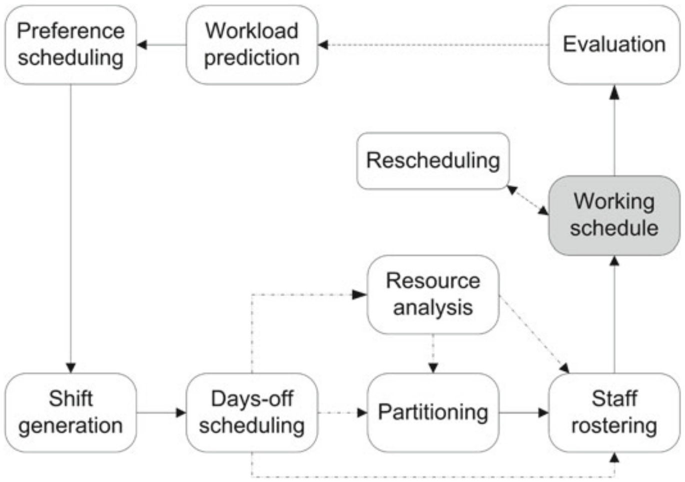

Fig. 1 The workforce scheduling process. The *upper boxes* represent the subphases that may occur in both mid-term and short-term planning (preprocessing phase). The subphases represented by the *lower boxes* (staff scheduling phase) only occur in short term planning, although information gathered from these may prove useful in future mid-term planning

made during previous subphases. This would be an untenable approach for finding the global optimum for most problems. However, our goal is to find a good enough solution for a broad range of problems. Our subphase-based approach is flexible enough to achieve this goal. Another benefit is decreased computational complexity due to the constantly narrowing search space.

The staff scheduling phase can be solved using computational intelligence. Computational workforce scheduling is key to increased productivity, quality of service, customer satisfaction and employee satisfaction. Other advantages include reduced planning time, reduced payroll expenses and ensured regulatory compliance.

3 The Preprocessing Phase

The preprocessing phase is the foundation upon which the actual staff scheduling phase is built. It may involve identifying both the needs of the customer(s) and the attributes (preferences, skills etc.) of the employees, and determining staffing requirements based on the former. This phase can be thought of as the transition between mid-term and short-term planning, since it touches both. This is the point in the workforce scheduling process where historical data and the schedules of previous planning horizons are most useful.

3.1 Workload Prediction/Determination

The nature of determining the amount and type of work to be done at any given time during the next planning horizon depends greatly on the nature of the job. If the workload is uncertain then some form of workload prediction is called for [8]. Some examples of this are the calls incoming to a call center or the customer influx to a hospital.

We define the *service level* SL(n) as the percentage of customers that need to wait for service for at most n seconds. Usually workload prediction aims to provide a certain service level (or above) for some fixed n. We simulate the randomly distributed workload based on historical data and statistical analysis, and find a suitable working employee structure (i.e. how many and what kinds of employees are needed) over time [11]. Computationally this approach is much more intensive than methods based on queuing theory. However, it has the benefit of being applicable to almost any real-world situation.

If the workload is static, no forecasting is necessary. For example, a local transport company might be under a strict contract to drive completely pre-assigned bus lines. In such a case shift generation may be necessary to combine the different bus lines into shifts, but the workload as such is static and thus calls for no forecasting.

3.2 *Preference Scheduling*

Research by Kellogg and Walczak [12] indicates that it is crucial for a workforce management system to allow the employees to affect their own schedules. In general it improves employee satisfaction. This in turn reduces sick leaves and improves the efficiency of the employees, which means more profit for the employer. Hence we use an easy-to-use user interface that allows the employees to input their preferences into the workforce management system. This eases the organizational workload of the personnel manager. A measure of fairness is incorporated via limiting the number and type of different wishes that can be expressed per employee. We are looking into incorporating more rigorous fairness measures and more complex preferences to the system. In our experience our current system is satisfactory, yet there is always room for improvement.

Preferences can be considered at every subphase of the staff scheduling phase [13–15]. The different types of preferences we consider are found under the respective subphases of the workforce scheduling process.

4 The Staff Scheduling Phase

4.1 *Shift Generation*

Shift generation transforms the determined workload into shifts. This includes deciding break times when applicable. Shift generation is essential especially in cases where the workload is not static. In other cases companies often want to hold on to their own established shift building methods.

A basic shift generation problem includes a variable number of activities for each task in each timeslot. Some tasks are not time-dependent; instead, there may be a daily quota to be fulfilled. Activities may require competences. The most important optimization target is to match the shifts to the workload as accurately as possible. In our solutions we create the shifts for each day separately, each shift corresponding to a single employee's competences and preferences. We do not minimize the number of different shifts. The choice between hard and soft constraints is given by the instances themselves.

We now present a real-world case from a Finnish haulage company. The problem as it was presented to us, related to a cargo terminal of theirs, is as follows. The planning horizon is five days, extending from a Sunday evening to a Friday evening. Each hour a number $M(d,h)$, where d = day and h = hour, of arrival manifests needs to be processed. The arrivals not handled immediately are queued, and the queue needs to be empty in the morning (6.30) and in the evening (20.30). The values of M for different days and hours can be found in [16].

It is assumed that the employees are identical in their processing capacity: each employee can handle 11 manifests per hour during the day (6–22) and 17.5 manifests per hour during the night (22–6). There are 76 full-time employees and 13 part-time employees. A full-time employee's shifts must be 4–10 h long, and the total working time over a 6-week period must be 240 h. A part-time employee's shifts must be 4–6 h long. Additionally, a part-time employee should have 3 shifts per week, and the total number of part-timers' working hours must be at most 10 % of the total working hours of all the employees.

The length of a timeslot is 30 min. The number of full-time employees was restricted to 69 in order to keep the workload of the part-time employees suitable. Thus we end up with 82 employees in total. The following hard constraints were used. The notation for different constraints is from [16].

(SGS1): No shift should contain timeslots with multiple types of activities.
(SGS2): No shift should contain gaps.
(SGP3): Each shift must contain a 30-min lunch break.
(SGV5): Each shift's length must be within the allowed limits of the corresponding employee.

The following soft constraints were used.

(SGP4): For shift s, let d_s be the distance of its break from the midpoint of the shift in slots (it does not matter if the optimal positions are not integers), and let a_s be 0.25 × (length of s). If $d_s > a_s$, then a cost of round($d_s - a_s$) is incurred.

(SGC4) + (SGC5): A compound constraint is used in order to make sure that the queue of arrivals is empty at 6.30 and 20.30, and that no working time is lost due to having too much workforce at work too early. The *cumulative effective workload* CEW[*day, type, time*] represents the workload that has effectively been contributed to handling the manifests up to timeslot *time*. It is calculated as

$$\mathrm{CEW}\,[day,\ type,\ time] = \min\left\{\begin{array}{l}\mathrm{MCS}\,[day,\ type,\ time]\,,\\ \mathrm{CEW}\,[day,\ type, time-1]\\ +\mathrm{w}\,[day,\ type,\ time]\end{array}\right\}, \tag{1}$$

where w is the number of workers scheduled to do a certain task at a certain time and MCS is the *maximum cumulative workload* given in [16]. For each day, the penalty given is the sum of differences between total workload and effective workload for day and night tasks.

The results are briefly described in Table 1 along with comparative numbers from the company's own solution. Our solution has no violations in SGP4, so the total penalty (649) represents exactly the number of timeslots that the effective working time is short of the total time that the jobs require ($649 \times 30 = 178,170 - 158,700$). The numbers from the company's current scheduling method made us doubt whether all the assumptions were close enough to reality and if the data/model were precise enough, but based on our results a contract for the use of our optimization software was signed.

Table 1 Comparison between our solution and a manual solution

	Our solution	Manual solution
Total working minutes (actual)	175,020	201,300
Effective working minutes (actual)	158,700	144,330
Total job minutes (goal)	178,170	178,170
Job completion (%)	89	81
Percentage of wasted working time	9	28

We hope to get more precise data in the future in order to improve both our model and our solutions. In Sect. 4.3.3 we'll roster the staff using the generated shifts.

4.2 Days-Off Scheduling

Days-off scheduling decides the rest days and the working days of the employees. It is based on the result of the shift generation: for each day a set of suitable employees must be available to carry out the shifts. This is the first subphase where employees' preferences usually have a big emphasis. The choice between hard and soft constraints is highly dependent on the problem instance.

We have used the list of constraints given in [16] to successfully model and solve some real-world days-off scheduling problems [13] and some nurse rostering problems [17].

4.3 Staff Rostering

4.3.1 Resource Analysis (Optional)

To see if there will be any chance of succeeding at matching the workforce with the shifts while adhering to the given constraints, an analysis is run on the data. If we have already optimized the days-off, this subphase is not necessary but it may still be useful. In addition to helping the personnel manager see the problem with the data quickly and efficiently, it may help convince the management level that the current practices and processes of generating the schedules are simply untenable. We have developed a statistical tool for this.

4.3.2 Partitioning of Massive Data (Optional)

Some real-world datasets are huge. They may consist of hundreds of employees with a corresponding number of jobs. Such datasets are often computationally very challenging. If there are no apparent "trivial" partitioning criteria (for example, a bus driver could be limited to driving buses starting from a specific bus depot, but if the constraint is not hard or there are drivers without a designated depot, it is not a trivial partitioning criterion) we can use the PEAST algorithm to partition the data, as in [14].

We now present a real-world case from a Finnish bus company. The problem consists of rostering 175 bus drivers over a planning horizon of 2 weeks. The days-off are invariant. There are 6 different kinds of days-off. The hard constraints of the problem are as follows. We have used the list of constraints given in [16] as the basis for modeling the case.

(SRR1): The working time of an employee must be strictly less than his/her goal working time. The shift time of an employee must be greater than his/her goal working time.

(SRR3): The rest time of 9 h must be respected between adjacent shifts.

(SRR5): There is 1 person with 6 working days during which he/she cannot work certain shifts.

(SRO3): The 3 most common kinds of days-off (90 % of all days-offs) must be whole, i.e. they cannot be immediately preceded by a shift that ends after midnight

(SRO4): There are in total 140 pre-assigned shifts.

The soft constraints of the problem are as follows.

(SRR1): The required number of working hours must be respected. The total working hours of the employees range from 1,200 to 4,815 min. The working minutes per day per employee range from 360 to 535. The difference (17,888 min) between employees' total working time goal (786,750 min) and the sum of the working time of all the shifts (768,862 min) should be evenly distributed among the shifts. There are 1,643 shifts, which results in approximately 10.9 min per shift. Each employee should thus be $SG(e) = 10.9\times$ (number of working days of employee e) minutes short of their personal workload goal. Define $S(e)$ as the actual shortage for employee e. If $|S(e) - SG(e)| > 0.1 \times SG(e)$, then the cost given is $|S(e) - SG(e)| - 0.1 \times SG(e)$. This ensures fairness in regard to the working time. The arbitrary threshold is used, since the goal is to have highly similar but not necessarily equal shortages.

Additionally, the linkage time (i.e. time spent having lunch or waiting for another vehicle, totalling 24,322 min in this instance) should be distributed evenly among the employees. This means approximately 14.8 min of linkage time per shift. Thus each employee should have $LG(e) = 14.8\times$ (number of working days of employee e) linkage minutes. Define $L(e)$ as the actual linkage time for employee e. If $|L(e) - LG(e)| > 0.1 \times LG(e)$, then the cost given is $|L(e) - LG(e)| - 0.1 \times LG(e)$. The linkage time is not nearly evenly distributed among different shift types. Almost 90 % of all linkage time belongs to the 60 % of shifts that start before 9 o'clock in the morning, which means that the early shifts have on average 6 times as much linkage

Table 2 Average and quartiles of 6 runs for total hard, total soft and preference constraint violations

	No partitions			Partitions		
	Hard	Soft	Pref	Hard	Soft	Pref
Average	11.7	383.8	284.5	2.5	499.0	224.7
Min	9.0	329.0	264.0	1.0	266.0	214.0
Q_1	10.3	343.8	268.5	2.0	373.5	223.5
Q_2	12.0	372.5	283.0	2.5	408.5	225.0
Q_3	13.0	423.8	297.5	3.0	586.0	228.0
Max	14.0	454.0	311.0	4.0	904.0	232.0

time as the later shifts. Since some employees only want morning shifts while others only want later shifts, compromises have to be made.

(SRR3): Each employee should have at least 11 h of rest time between two adjacent shifts. Each violation of this rule incurs a cost of 1.

(SRP2): There are 1,102 working days with a shift type preference defined. Each unfulfilled wish incurs a cost of 1.

Our results both using and not using partitioning are briefly described in Table 2. One hard constraint violation is unavoidable: there is an employee whose previous planning horizon ended with a late job, yet he has an early pre-assigned job on the first Monday of the new planning horizon, causing a rest time violation. This is a very challenging dataset and as such it shows that partitioning has its benefits. However, in order to eliminate the remaining hard constraint violations with consistency we need to either consider alternative methods or, as the preferred alternative, point out to the problem owner the inaccuracies in their current system and investigate what could be done to rectify the problems caused by their contradictory constraints.

4.3.3 Staff Rostering (Shift Scheduling)

The final optimization subphase of the workforce scheduling process is staff rostering, during which the shifts are assigned to the employees. The length of the planning horizon for this subphase is usually between two and six weeks. The preferences of the employees are usually given a relatively large weight but, as before, the choice between hard and soft constraints stems from the instances themselves. The most important constraints are usually resting times and certain competences, since these are often laid down by the collective labour agreements and government regulations. Working hours of the employees are also important. We have used the list of constraints given in [16] to successfully model and solve some real-world staff rostering cases [14, 18] along with some nurse rostering cases [17].

In Sect. 4.1 we generated the shifts for a haulage company. Next we will schedule those shifts in order to optimize working time and resting time for each employee. In this case no separate days-off scheduling is necessary, since there are no constraints involving days-off directly.

We generated shifts for 69 full-time employees and 13 part-time employees. A full-time employee's shifts must be 4–10 h long, and the total working time over a 6-week period must be 240 h. However, our planning horizon is only 5 days (one week), so each full-timer should have approximately 40 h of work. A part-time employee's shifts must be 4–6 h long, and a part-time employee should have 3 shifts per week. The following hard constraints were used. The notation for different constraints is from [16].

(SRR3): Each employee must have at least 7 h of rest time between two adjacent shifts.
(SRO1): Part-timers only have competence to work shifts that are less than 6 h in length.

The following soft constraints were used.

(SRR1): Each full-time employee should have a total working time of 2,400 min. Each part-time employee should work 3 shifts.
(SRR3): Each employee should have at least 11 h of rest time between two adjacent shifts. Each violation of this rule incurs a cost of 1.

We scheduled 52 full-time employees with a total working time of 2,400 min and 17 full-time employees with a total working time of 2,370 min, which is optimal. There are 13 violations in the rest time constraint (SRR3). Every part-timer has 3 shifts. Thus the schedule is acceptable.

5 Our Solution Method

The PEAST algorithm [19, 20] is a population-based local search method. The acronym PEAST stems from the methods used: Population, Ejection, Annealing, Shuffling and Tabu. Aside from workforce scheduling, it has been used to solve real-world school timetabling problems [21] and real-world sports scheduling problems [22]. The PEAST algorithm uses GHCM, the Greedy Hill-Climbing Mutation heuristic introduced in [23] as its local search method. The pseudo-code of the algorithm is given in Fig. 2.

The reproduction phase of the algorithm is, to a certain extent, based on steady-state reproduction: the new schedule replaces the old one if it has a better or equal objective function value. Furthermore, the least fit is replaced with the best one when n better schedules have been found, where n is the size of the population. Marriage selection is used to select a schedule from the population of schedules for a single GHCM operation. In the marriage selection we randomly pick a schedule, S, and then attempt to randomly pick a better one at most $k - 1$ times. We choose the first better schedule, or, if none is found, we choose S.

```
Set the time limit $t$, no_change limit $m$ and the population size $n$
Generate a random initial population of individuals
Set *no_change* = 0 and *better_found* = 0
WHILE elapsed_time < $t$
REPEAT $n$ times
    Select an individual $A$ by using a marriage selection with $k = 3$
    (explore promising areas in the search space)
    Apply GHCM to $A$ to get a new individual $A'$
    Calculate the change $\Delta$ in objective function value
    IF $\Delta <= 0$ THEN
        Replace $A$ with $A'$
        IF $\Delta < 0$ THEN
            better_found = better_found + 1
            no_change = 0
            END IF
       ELSE
            no_change = no_change + 1
    END IF
END REPEAT
IF better_found > $n$ THEN
    Replace the worst individual with the best individual
    Set better_found = 0
END IF
IF no_change > $m$ THEN
    (escape from the local optimum)
    Apply shuffling operators
    Set no_change = 0
END IF
 (avoid staying stuck in the promising search areas too long)
Update simulated annealing framework
Update the dynamic weights of the hard constraints (ADAGEN)
END WHILE
Choose the best individual from the population
```

Fig. 2 The pseudo-code of the PEAST algorithm

The heart of the GHCM heuristic is based on similar ideas to the Lin-Kernighan procedures [24] and ejection chains [25]. The basic hill-climbing step is extended to generate a sequence of *moves* in one step, leading from one solution candidate to another. The GHCM heuristic moves an *object*, o_1, from its old position in some *cell*, c_1, to a new cell, c_2, and then moves another object, o_2, from cell c_2 to a new cell, c_3, and so on, ending up with a sequence of moves. An object is a task-based activity or a whole break (in shift generation), a day-off (in days-off scheduling) or a shift (in shift scheduling). A cell is a shift (in shift generation) or an employee (in days-off scheduling and shift scheduling). A move involves removing an object from a certain position within a cell and inserting it either into a new cell (position is invariant) or a new position (cell is invariant).

The initial cell selection is random. The cell that receives an object is selected by considering all the possible cells and selecting the one that causes the least increase in the objective function when only considering the relocation cost. Then, another object from that cell is selected by considering all the objects in that cell and picking the one for which the removal causes the biggest decrease in the objective function when only considering the removal cost. Next, a new cell for that object is selected, and so on. The sequence of moves stops if the last move causes an increase in the objective function value and if the value is larger than that of the previous non-improving move. Then, a new sequence of moves is started. The initial solution is randomly generated.

The decision whether or not to commit to a sequence of moves in the GHCM heuristic is determined by a refinement [23] of the standard simulated annealing method [26]. Simulated annealing is useful to avoid staying stuck in the promising search areas for too long. The initial temperature T_0 is calculated by

$$T_0 = 1/\log(1/X_0) \tag{2}$$

where X_0 is the degree to which we want to accept an increase in the cost function (we use a value of 0.75). The exponential cooling scheme is used to decrement the temperature:

$$T_k = \alpha T_{k-1} \tag{3}$$

where α is usually chosen between 0.8 and 0.995. We stop the cooling at some predefined temperature. Therefore, after a certain number of iterations, m, we continue to accept an increase in the cost function with some constant probability, p. Using the initial temperature given above and the exponential cooling scheme, we can calculate the value

$$\alpha = (-1/(T_0 \log p))^{1/m}. \tag{4}$$

We choose m equal to the maximum number of iterations with no improvement to the cost function and p equal to 0.0015.

For most PEAST applications we introduce a number of shuffling operators—simple heuristics used to perturb a solution into a potentially worse solution in order to escape from local optima—that are called upon according to some rule. The most used heuristics include moving a single random object from one cell to another random cell, or swapping two random objects between two random cells. For further details on the different shuffling operators used, see [13–15], [17, 18, 27]. The operator is called every $l/20$th iteration of the algorithm, where l equals the maximum number of iterations with no improvement to the cost function.

We use the weighted-sum approach for multi-objective optimization. We use the ADAGEN method [23] which assigns dynamic weights to the hard constraints. The weights are updated every kth generation using the formula given in [23]. The soft constraint weights are static yet instance-dependent.

References

1. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York
2. Tien J, Kamiyama A (1982) On manpower scheduling algorithms. SIAM Rev 24(3):275–287
3. Lau HC (1996) On the complexity of manpower shift scheduling. Comp Oper Res 23(1):93–102
4. Marx D (2004) Graph coloring problems and their applications in scheduling. Periodica Polytech Ser El Eng 48:5–10
5. Di Gaspero L, Gärtner J, Kortsarz G, Musliu N, Schaerf A, Slany W (2007) The minimum shift design problem. Ann Oper Res 155(1):79–105
6. Dantzig GB (1954) A comment on Edie's traffic delays at toll booths. Oper Res 2:339–341
7. Alfares HK (2004) Survey, categorization and comparison of recent tour scheduling literature. Ann Oper Res 127:145–175
8. Ernst AT, Jiang H, Krishnamoorthy M, Sier D (2004) Staff scheduling and rostering: a review of applications, methods and models. Eur J Oper Res 153(1):3–27
9. Meisels A, Schaerf A (2003) Modelling and solving employee timetabling problems. Ann Math Artif Intell 39:41–59
10. De Causmaecker P, Vanden Berghe G (2012) Towards a reference model for timetabling and rostering. Ann Oper Res 194(1):167–176
11. Nurmi K, Kyngäs N, Salli J (2012) A workload prediction, staffing and shift generation method for contact centers. In: Proceedings of the 4th EURO working group on stochastic modeling, Paris, France
12. Kellogg DL, Walczak S (2007) Nurse scheduling: from academia to implementation or not? Interfaces 37(4):355–369
13. Kyngäs J, Nurmi K (2011) Days-off scheduling for a bus transportation company. Int J Innovative Comput Appl 3(1):42–49
14. Kyngäs N, Nurmi K, Kyngäs J (2012) Optimizing large-scale staff rostering instances. In: Proceedings of the international multiconference of engineers and computer scientists, Hong Kong. Lecture notes in engineering and computer science, pp 1524–1531
15. Kyngäs N, Goossens D, Nurmi K, Kyngäs J (2012) Optimizing the unlimited shift generation problem. In: Proceedings of the international conference on the applications of evolutionary computation, Malaga, Spain, pp 508–518
16. Kyngäs N, Nurmi K, Kyngäs J (2013) The workforce scheduling process using the PEAST algorithm. In: Proceedings of the international multiconference of engineers and computer scientists, 13–15 Mar 2013, Hong Kong. Lecture notes in engineering and computer science, pp 1048–1056
17. Kyngäs N, Nurmi K, Ásgeirsson EI, Kyngäs J (2012) Using the PEAST algorithm to roster nurses in an intensive-care unit in a Finnish hospital. In: Proceedings of the 9th conference on the practice and theory of automated timetabling (PATAT), Son, Norway

18. Nurmi K, Kyngäs J, Post G (2012) Driver rostering for a Finnish transportation company. In: Ao Sio-Iong (ed) IAENG transactions on engineering technologies, vol 7. Springer, USA
19. Kyngäs J (2011) Solving challenging real-world scheduling problems. Ph.D. dissertation, Department of Information Technology, University of Turku, Finland. Available: http://urn.fi/URN:ISBN:978-952-12-2634-2
20. Kyngäs N, Nurmi K, Kyngäs J (2013) Crucial components of the PEAST algorithm in solving real-world scheduling problems. In: 2nd International conference on software and computer applications (ICSCA 2013) (submitted for publication)
21. Nurmi K, Kyngäs J (2007) A framework for school timetabling problem. In: Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications. France, Paris, pp 386–393
22. Kyngäs J, Nurmi K (2009) Scheduling the Finnish major ice hockey league. In: Proceedings of the IEEE symposium on computational intelligence in scheduling. Nashville, USA
23. Nurmi K (1998) Genetic algorithms for timetabling and traveling salesman problems, Ph.D. dissertation, Department of Applied Mathematics, University of Turku, Finland. Available: http://www.bit.spt.fi/cimmo.nurmi/dissertation/cimmodis.zip
24. Lin S, Kernighan BW (1973) An effective heuristic for the traveling salesman problem. Oper Res 21:498–516
25. Glover F (1992) New ejection chain and alternating path methods for traveling salesman problems. In: Sharda, Balci, Zenios (eds) Computer science and operations research: new developments in their interfaces. Elsevier, pp 449–509
26. van Laarhoven PJM, Aarts EHL (1987) Simulated annealing: theory and applications. Kluwer Academic Publishers
27. Kyngäs N, Nurmi K, Kyngäs J (2013) Solving the person-based multitask shift generation problem with breaks. In: 5th International conference on modeling, simulation and applied optimization (ICMSAO'13) (submitted for publication)